


# Structuring Data with XML

---

## Objectives

- ▶ Understand eXtensible Markup Language (XML)
- ▶ Define XML elements and structure
- ▶ Enter XML data
- ▶ Bind XML data to HTML
- ▶ Format XML data with HTML
- ▶ Display XML data with HTML
- ▶ Modify an XML document
- ▶ Alter XML data view with HTML

HTML has evolved from a way of structuring information into a language for controlling the format, or display, of Web content. With the vast growth of data on the Web, and the creation of many new applications, it has become critical to have a standard but expandable means to define, structure, and exchange the data on the Web. The **eXtensible Markup Language (XML)** is designed to ensure a **universal data structure** that can be read by any XML-compliant browser yet still allow Web designers the freedom to create **custom data definitions** for an endless variety of applications (e.g., a book or movie database).  Lydia would like to create a “Recommended Backpacking Books” page for backpacking enthusiasts on the Nomad Web site. The HTML page will display book data she stores using XML and custom data definitions.



# Understanding eXtensible Markup Language (XML)

**XML (eXtensible Markup Language)** is a text-based syntax especially designed to describe, deliver, and exchange structured data. XML documents use the file extension .xml and, like HTML files, can be created with a simple text editor. XML is not meant as a replacement for HTML but, rather, as a means to vastly extend its descriptive and structural power. XML uses a syntax that is similar to HTML, with four basic differences. These rules guarantee that a document is well-formed. A **well-formed** document requires that data be uniformly structured by tagsets so it can be read correctly by any XML-compliant program. Lydia is new to using XML, so she researches the language before starting to create the backpacking book list. To understand how the syntax of XML differs from HTML, Lydia examines each XML syntax rule carefully.

## Details



### All elements must have start and end tags

An **element** consists of the start tag and corresponding end tag along with the content between the tagset. Unlike HTML, where you can mark up a document without using some closing tags (e.g., `<P>`), XML requires that both opening and closing tags be present. For example, the first line in Figure N-1 shows the correct way to code XML with both the start and end paragraph tags necessary to create a well-formed XML document. The same line in Figure N-2 lacks the required closing `</P>` tag, and thus violates the first rule of XML syntax.



### All elements must be nested correctly

Just as in HTML, when writing XML, you should close first whatever tagset you opened last. However, you cannot overlap elements in XML. The second line of code in Figure N-1 shows the heading tagset `<H2></H2>` correctly nested inside of the paragraph tagset `<P></P>`. The same line in Figure N-2 breaks this rule by placing the closing `</H2>` tag after the end paragraph tag `</P>`. Although this code would display properly in HTML, it is not well-formed and would not display in XML because of the inherent stricter rule enforcement.



### All attribute values must appear with quotation marks

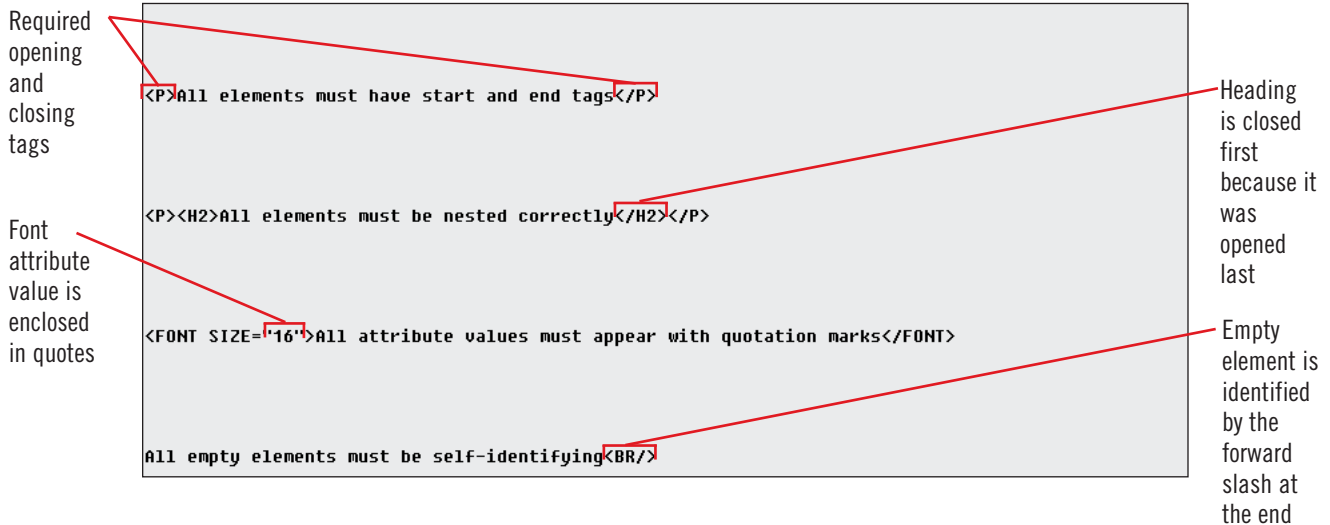
HTML requires that only certain attribute values, such as URLs and strings, be in quotes. Values such as image and font size may be used without quotes. In XML, attributes must appear in quotes. The third line of code in Figure N-1 illustrates the proper way to code by quoting the font size attribute value `"16"`, whereas the same line in Figure N-2 shows the font attribute value without quotes and, thus, wrongly marked up for XML.



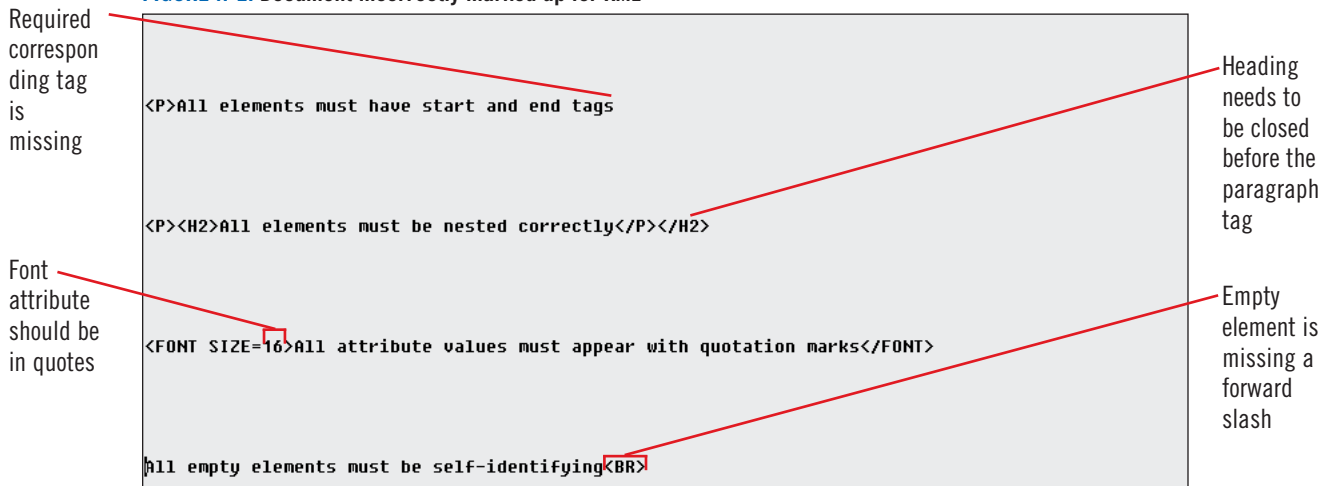
### All empty elements must be self-identifying by ending with `</>`

An **empty element** is one that doesn't have a closing tag (e.g., `<BR>`, `<HR>`, and `<IMG>` in HTML). The last tag on the fourth line of code in Figure N-1 demonstrates the proper way to identify an empty element in XML by ending it with `</>`. This forward slash at the end of the tag indicates that the element, or container, is not broken (i.e., missing a closing tag) but simply empty. The same tag in Figure N-2 does not contain the necessary ending slash and, hence, would prevent an XML document from being well-formed.

**FIGURE N-1: Document correctly marked up for XML**



**FIGURE N-2: Document incorrectly marked up for XML**



## Benefits of XML

With XML, it is possible to identify data in meaningful ways, much like a database lets you identify and organize data with unique fieldnames and records. XML allows custom “vocabularies,” or element sets, to be defined for particular types of data, such as books, movies, auto parts, legal cases, and medical information. In other words, these custom vocabularies act like the fieldnames in a conventional database to clearly identify and segment data. If these highly descriptive elements come into widespread use on the Web, XML has the potential to organize the Web into a coherent body of knowledge. For instance, this new level of semantics should improve the ability of now-overburdened search engines to rapidly find relevant information—in the same way queries can quickly

locate relevant data in a database. Additionally, XML offers the means to exchange and process data from otherwise-incompatible information repositories on the Internet. XML and DHTML also enable a significant portion of the processing load to be shifted from Web servers to Web browsers (clients). Consequently, applications that require high-level compatibility and performance, such as those in electronic commerce, will greatly benefit from the implementation of XML. Finally, intelligent Web applications that seek out choice bits of information on the Web to match the preferences of individual users also should realize equally significant gains in accuracy as a result of clearly labeled XML data.



# Defining XML Elements and Structure

Unlike HTML, XML is not confined to a restricted library of tagsets. XML gives you the freedom to define a limitless set of custom elements to fit any application or situation. Also, XML enables you to organize data into a **tree-like hierarchical structure** by nesting elements within other elements. For example, Figure N-3 shows an XML document with a set of elements designed to precisely describe the type of data to be stored (e.g., Book, Name, Author, etc.). In addition, by nesting the **child** elements Name, Author, and Publisher within the **parent** element Book, and nesting the Book elements within BookList, you automatically establish a parent-child, or hierarchical, structure similar to the one illustrated in Figure N-4. After completing her basic research on XML, Lydia decides to define the XML elements and structure necessary to store the data in the backpacking book list.

## Steps 1 2 3 4

1. Open your text editor, then type `<?XML VERSION='1.0'?>`

The beginning of an XML document, called the **prolog**, contains the **XML declaration**, which is a processing instruction for the browser or other XML-compliant program reading the document. This processing instruction begins with the opening delimiter `<?` and ends with the closing delimiter `?>`. The instruction included between the delimiters, `XML VERSION='1.0'`, indicates that the document should be read as an XML file.

### QuickTip

The indentation shown here is unnecessary for proper processing but helps to more clearly delineate the relationship between the document elements.

2. Press **[Enter]** twice, then type the following elements with the appropriate indentations:

```
<BookList>
  <Book>
    <Name></Name>
    <Author></Author>
    <Publisher></Publisher>
  </Book>
</BookList>
```

The elements in an XML document are **ranked** from the outermost set to the innermost elements. The highest ranked element forms the “root” of a tree-like structure, while the lower ranked elements make up the successive “branches” of the tree, as shown in Figure N-4. Thus, the rank of an element determines how much of the tree it controls.

3. To create two more instances of the Book elements, begin by selecting the highlighted area shown in Figure N-5
4. Press **[Ctrl][c]** to copy the highlighted area

### QuickTip

The prefix “XML” is reserved for XML syntax, so you can’t use it as a prefix for tag name (e.g., XMLbook).

5. Place the insertion point at the end of the Book element `</Book>`, then press **[Enter]** twice

6. Press **[Ctrl][v]** to paste a second Book record

A second instance of the Book elements appears in your document.

### QuickTip

XML is case-sensitive. For example, the element `<Author></Author>` is not equivalent to the element `<AUTHOR></AUTHOR>`.

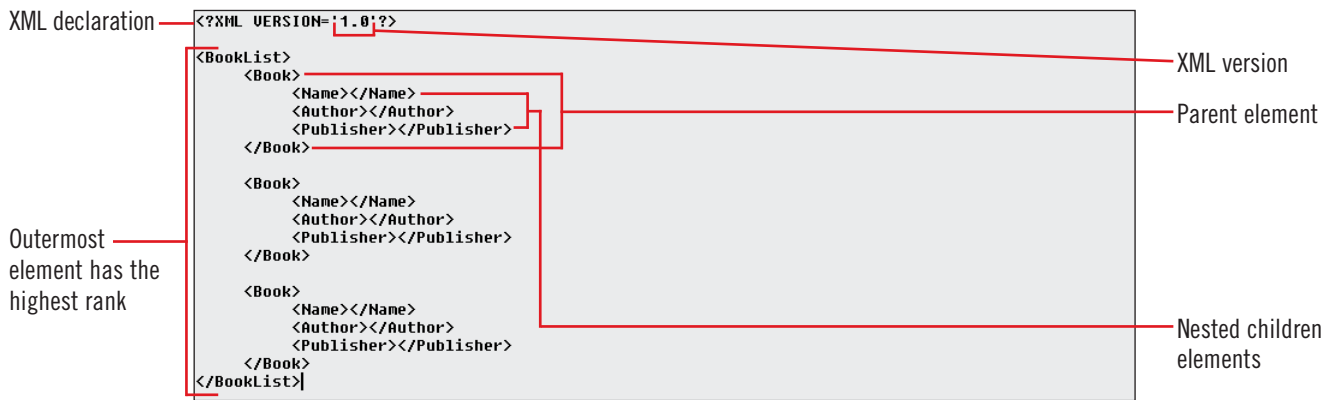
7. Press **[Enter]** once, paste another instance of the Book elements, then press **[Delete]**  
Your document should now look like the one shown in Figure N-3.

8. Save the file as a text document with the filename **books.xml**

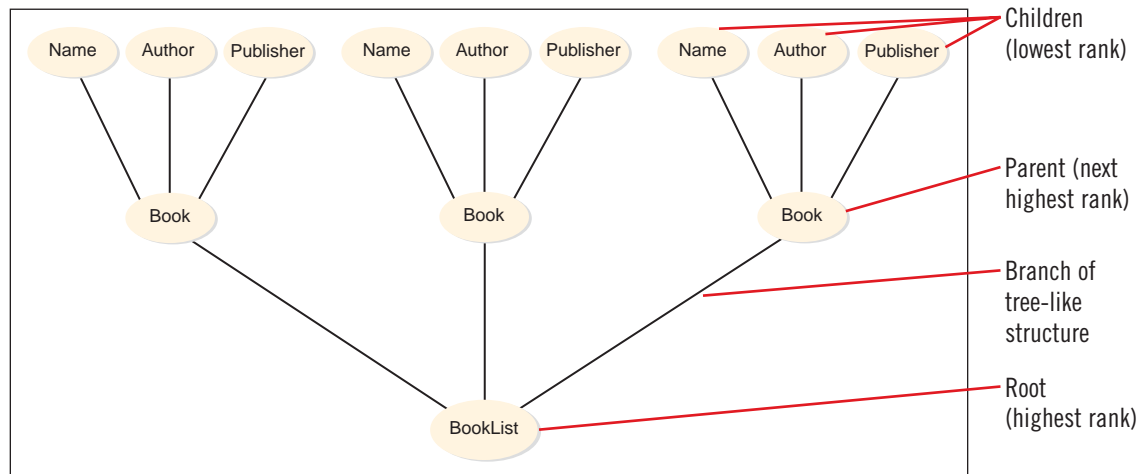
You are now ready to enter the data into your XML document.



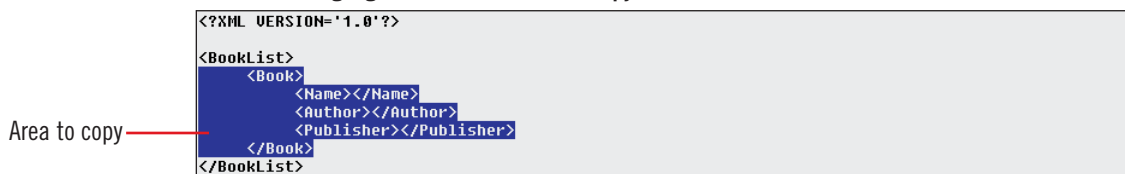
**FIGURE N-3: Elements and structure of book list XML document**



**FIGURE N-4: Tree-like hierarchical structure of XML document**



**FIGURE N-5: Highlighted book elements to copy**




## Document Type Definition (DTD) and XML schemas

A **Document Type Definition (DTD)** is the formal specification of the rules of an XML document—namely, which elements are allowed and in what combinations. A Web page designer would create a DTD to ensure that any XML document using the DTD will be **valid** (i.e., comply with the formal specification). For example, a DTD could be set up to ensure that all XML files dealing with drug prescriptions include certain data elements (e.g., the prescribing doctor's name, expiration date, and refill information). Although a DTD can be called from an XML file, it uses a different syntax from XML. An

**XML schema** combines the concepts of a DTD, relational databases, and object-oriented designs to create a richer and more powerful way to formally define the elements and structure of an XML document. In addition, XML schemas use the same syntax as XML, so they will be able to appear in the same document. However, support for XML schemas are not supported in Internet Explorer 4 and DTDs only enjoy limited support. To learn more about DTDs and XML schemas, visit <http://www.microsoft.com/xml/>, or search the <http://www.microsoft.com> for information on XML.



# Entering XML Data

You enter data into an XML document the same way you do an HTML page—either manually with an editor or automated with a sophisticated HTML form.  Lydia wants to enter data about three backpacking handbooks in the XML file to test her knowledge of how this new technology works. She decides to use a text editor to enter the data.

## Steps 1 2 3 4

### QuickTip

All the content of an XML element is treated as data, including white space. Make sure not to include unwanted blank space in between any opening and closing tagsets.

1. Make sure that **books.xml** is open in your text editor
2. Position the insertion point between the first set of **<Name></Name>** tagsets, as shown in Figure N-6
3. Type **The Backpacker's Field Manual**
4. Place the insertion point between the **<Author></Author>** tagset directly below, then type **Rick Curtis**
5. Move the insertion point between the **<Publisher></Publisher>** tagset directly below, then type **Crown Publishers Inc.**  
The text editor should match the contents of Figure N-7.
6. Type the following data in the appropriate tagsets for the next two instances of the Book elements:

**The Modern Backpacker's Handbook**

**Glenn Randall**

**Lyons and Burford Publishers**

**The Backpacker's Handbook**

**Chris Townsend**

**Ragged Mountain Press**

The text editor window should look like the one shown in Figure N-8.

7. Save, then close the file

FIGURE N-6: Insertion point position

```
<?XML VERSION='1.0'?>
<BookList>
  <Book>
    <Name>X</Name>
    <Author>Y</Author>
    <Publisher>Z</Publisher>
  </Book>
  <Book>
    <Name></Name>
    <Author></Author>
    <Publisher></Publisher>
  </Book>
  <Book>
    <Name></Name>
    <Author></Author>
    <Publisher></Publisher>
  </Book>
</BookList>
```

Correct insertion  
point position

FIGURE N-7: XML document with first data record

```
<?XML VERSION='1.0'?>
<BookList>
  <Book>
    <Name>The Backpacker's Field Manual</Name>
    <Author>Rick Curtis</Author>
    <Publisher>Crown Publishers Inc.</Publisher>
  </Book>
  <Book>
    <Name></Name>
    <Author></Author>
    <Publisher></Publisher>
  </Book>
  <Book>
    <Name></Name>
    <Author></Author>
    <Publisher></Publisher>
  </Book>
</BookList>
```

Data record

FIGURE N-8: XML document with all data entered

```
<?XML VERSION='1.0'?>
<BookList>
  <Book>
    <Name>The Backpacker's Field Manual</Name>
    <Author>Rick Curtis</Author>
    <Publisher>Crown Publishers Inc.</Publisher>
  </Book>
  <Book>
    <Name>The Modern Backpacker's Handbook</Name>
    <Author>Glenn Randall</Author>
    <Publisher>Lyons and Burford Publishers</Publisher>
  </Book>
  <Book>
    <Name>The Backpacker's Handbook</Name>
    <Author>Chris Townsend</Author>
    <Publisher>Ragged Mountain Press</Publisher>
  </Book>
</BookList>
```

Data  
record 1

Data  
record 2

Data  
record 3



# Binding XML Data to HTML

XML-compliant browsers such as Internet Explorer 4 support XML by including an XML parser and an XML Data Source Object (XML DSO). An **XML parser** dissects and interprets XML elements, whereas the **XML DSO** enables binding of the XML data to the HTML document using the DHTMLObject Model. Thus, the parser and DSO cooperate to allow the display, or **rendering**, of XML data in HTML. Lydia wants to display the data in the books.xml file in an HTML document that will become part of Nomad Ltd's Web site. Before she can format the data, Lydia must first create the HTML document and bind the XML data to it.

## Steps 1 2 3 4

1. Open your text editor, then type

```
<HTML>
<HEAD>
    <TITLE>Backpacking Book List</TITLE>
</HEAD>
<BODY>
    <H2>Recommended Backpacking Books</H2>
```

The text editor screen now should look like the one shown in Figure N-9. Lydia is ready to enter the XML DSO Java applet necessary to bind the XML data you created in the last lesson. A **Java applet** is a small program written in the Java programming language that is summoned using the <applet></applet> tagset.

2. Press **[Enter]** twice, then type

```
<APPLET CODE="com.ms.xml.dso.XMLDSO.class" WIDTH="100%" HEIGHT="25"
ID="xmldso" MAYSCRIPT="true">
    <PARAM NAME="url" VALUE="books.xml">
</APPLET>
```

The document should appear as shown in Figure N-10. Notice that the opening applet tag causes the browser to call the XML DSO Java applet. The parameter tag "<PARAM NAME='url' VALUE='books.xml'>" specifies a URL with the name of the XML file to bind to this HTML document.

3. Press **[Enter]**, then type:

```
</BODY>
</HTML>
```

With these closing HTML tags entered, the text editor screen should now match Figure N-11.

4. Save the file as a text document with the filename **books.htm**

Now Lydia has created an HTML document to which the XML data entered in the previous lesson is bound. The next step is to construct a table to format and display the bound data in your HTML document.



FIGURE N-9: Beginning HTML markup

```
<HTML>
<HEAD>
  <TITLE>Backpacking Book List</TITLE>
</HEAD>
<BODY>
<H2>Recommended Backpacking Books</H2>
```

FIGURE N-10: Applet to call the XML DSO

```
<HTML>
<HEAD>
  <TITLE>Backpacking Book List</TITLE>
</HEAD>
<BODY>
<H2>Recommended Backpacking Books</H2>

<APPLET CODE="com.ms.xml.dso.XMLDSO.class" WIDTH="100%"
HEIGHT="25" ID="xmldso" MAYSCRIPT="true">
  <PARAM NAME="url" VALUE="books.xml">
</APPLET>
```

Parameter name that specifies source as a URL

Value that specifies XML document filename

Applet to bind XML document to this HTML page

FIGURE N-11: HTML document with data-binding code

```
<HTML>
<HEAD>
  <TITLE>Backpacking Book List</TITLE>
</HEAD>
<BODY>
<H2>Recommended Backpacking Books</H2>

<APPLET CODE="com.ms.xml.dso.XMLDSO.class" WIDTH="100%"
HEIGHT="25" ID="xmldso" MAYSCRIPT="true">
  <PARAM NAME="url" VALUE="books.xml">
</APPLET>
</BODY>
</HTML>
```



# Formatting XML Data with HTML

Once the data in an XML document has been bound to an HTML page, you can use all of the available formatting capabilities in HTML (plus CSS) to control its presentation in a browser. Because XML data often consists of a list, or database, the table feature in HTML is an ideal vehicle for displaying this tabular data. In addition to the conventional table attributes, several new HTML attributes enable Web page designers to control the binding of XML data to their documents. Table N-1 describes these new attributes. Lydia decides to use a table to format and display her book-list data in the XML file.

## Steps 1234

### QuickTip

You can use all the power and flexibility of DHTML to manipulate XML data once it appears in an HTML document.

1. Make sure the file **books.htm** is open in your text editor
2. Place the insertion point below the **</APPLET>** tag, then press **[Enter]**  
Lydia wants to use a table to display the data in the books.xml file.
3. Carefully type the following:

```
<TABLE ID="table" BORDER="2" WIDTH="100%" DATASRC="#xmldso"
CELLPADDING="5">
<THEAD>
<FONT FACE="Arial" SIZE="2">
<TR>
<TH>TITLE</TH>
<TH>AUTHOR</TH>
<TH>PUBLISHER</TH>
</TR>
</FONT>
</THEAD>
<FONT FACE="Times New Roman" SIZE="2">
<TR>
<TD VALIGN="top"><DIV DATAFLD="NAME"
DATAFORMATAS="HTML"></DIV></TD>
<TD VALIGN="top"><DIV DATAFLD="AUTHOR"
DATAFORMATAS="HTML"></DIV></TD>
<TD VALIGN="top"><DIV DATAFLD="PUBLISHER"
DATAFORMATAS="HTML"></DIV></TD>
</TR>
</FONT>
</TABLE>
```

The text editor screen should match Figure N-12. This code creates a table that uses an Arial font style, with a size of 2, to display the headings TITLE, AUTHOR, and PUBLISHER across the top row. The DATAFLD attributes in this code specify the element from the XML document to be bound to each column. The DATAFORMATAS attributes indicate that the XML-based data should be displayed in HTML format. In other words, the DATAFORMATAS attributes tell your browser to interpret and format the imported XML data as HTML content. This code will create as many rows as necessary to display all the data (records) in your XML file.

4. Check your work to make sure your typing was completely accurate
5. Save, then close the file

**FIGURE N-12:** Table code to format and bind XML data

```

<HTML>
<HEAD>
  <TITLE>Backpacking Book List</TITLE>
</HEAD>
<BODY>
  <H2>Recommended Backpacking Books</H2>

  <APPLET CODE="com.ms.xml.dso.XMLDSO.class" WIDTH="100%"
  HEIGHT="25" ID="xmldso" MAYSCRIPT="true">
    <PARAM NAME="url" VALUE="books.xml">
  </APPLET>

  <TABLE ID="table" BORDER="2" WIDTH="100%"
  DATASRC="#xmldso" CELLSPACING="5">
    <THEAD>
      <FONT FACE="Arial" SIZE="2">
        <TR>
          <TH>TITLE</TH>
          <TH>AUTHOR</TH>
          <TH>PUBLISHER</TH>
        </TR>
      </FONT>
    </THEAD>
    <FONT FACE="Times New Roman" SIZE="2">
      <TR>
        <TD VALIGN="top"><DIV DATAFLD="NAME"
        DATAFORMATAS="HTML"></DIV></TD>
        <TD VALIGN="top"><DIV DATAFLD="AUTHOR"
        DATAFORMATAS="HTML"></DIV></TD>
        <TD VALIGN="top"><DIV DATAFLD="PUBLISHER"
        DATAFORMATAS="HTML"></DIV></TD>
      </TR>
    </FONT>
  </TABLE>

```

Font settings for table headings

Font settings for table rows

DATAFORMATAS attribute indicates the bound data should be displayed as HTML

DATAFLD attribute specifies the AUTHOR element be bound to this column

Table column heading for XML data

**TABLE N-1:** New table data-binding HTML attributes

attribute	description
DATASRC	Identifies the XML DSO applet used to bind the data.
DATAFLD	Specifies the particular column to bind the element to.
DATAFORMATAS	Indicates how the bound data should be rendered in the specified column (e.g., in HTML).
DATAPAGESIZE	Controls how many records are displayed in a table at once.




## Formatting XML data with the Extensible Style Language (XSL)

In addition to formatting XML data with HTML and CSS, you also can use an XSL (Extensible Style Language) stylesheet. An **XSL stylesheet** is a set of programming rules that determine how XML data is displayed in an HTML document. Because XSL is designed to work with XML, it has the same flexibility and syntax as XML. However, native support for XSL is not built into Internet Explorer 4 or other browsers. To apply an XSL stylesheet to an XML

document using Internet Explorer 4, first you must download and install the **Microsoft XSL ActiveX control**. This control is based on Microsoft's object-oriented ActiveX technology and is freely distributed. In fact, you can embed a script in your HTML document to automate the downloading and installation process. For more information on XSL and the XSL ActiveX control, see <http://www.microsoft.com/xml/>.



# Displaying XML Data with HTML

Once you have created an HTML document to bind and format your XML-based data, you simply need to start your browser and open the HTML page to view the results. The XML document storing the data and the HTML page presenting it can reside on your local computer, network, or a remote Web server on the Internet.  Lydia wants to see the XML data displayed in her HTML document.

## Steps 1 2 3 4

### QuickTip

Use Internet Explorer 4 to display XML data with HTML. Netscape Navigator 4 doesn't support XML, Navigator 5 will.

### Trouble?

If no error appears, yet the table fails to display correctly, then your installation of Internet Explorer may be missing the Java VM (Virtual Machine). The Java VM is available from the Internet Explorer downloads section at the Microsoft Web site. See your instructor or technical support person for assistance.

### 1. Start your browser, then open the **books.htm** file

The document shown in Figure N-13 should appear, complete with the formatted data read from your books.xml file. Notice the colored line above the table indicating that the XML was successfully loaded: "file:/A:/books.xml." If the load was unsuccessful, this line indicates an error in parsing the XML file.

### 2. If your HTML document displays an error when attempting to load the books.xml file like the one shown in Figure N-14, you need to edit your XML document, recheck your typing, and fix any syntax mistakes

Unlike HTML, an XML document won't load correctly unless you obey all the rules of a well-formed document.

### 3. After successfully viewing the book list in the HTML table, close your browser

**FIGURE N-13:** HTML document displaying formatted XML data in a table

Column headings

Rows of XML data

## Recommended Backpacking Books

Successfully loaded XML from "file:///A:/books.xml"

TITLE	AUTHOR	PUBLISHER
The Backpacker's Field Manual	Rick Curtis	Crown Publishers Inc.
The Modern Backpacker's Handbook	Glenn Randall	Lyons and Burford Publishers
The Backpacker's Handbook	Chris Townsend	Ragged Mountain Press

**FIGURE N-14:** HTML document displaying a parsing error

Highlighted line indicates nature of the problem

## Recommended Backpacking Books

Error loading XML document 'books.xml': com.ms.xml.parser.ParseException: Close tag BOOK does not match start tag PUBLISHER

TITLE	AUTHOR	PUBLISHER
The Backpacker's Field Manual	Rick Curtis	Crown Publishers Inc.



### Multiple views of data


Once data has been moved into your browser, it can be displayed in many different ways. A Web page designer might build several different views of the same data depending on the audience. For example, in the case of a movie database, the average user might just want to see the title, actors, and general plot of films, whereas some devoted fans will want to view all the particulars like the date the movie was

released, who directed it, and so forth. Because XML only describes data, not its appearance, a Web page designer is free to use HTML/CSS to create unique views of the data for different classes of users. In addition, with use of DHTML, the view of XML data can be manipulated easily by the end user to suit his or her needs and tastes.





# Modifying an XML Document

You can change an XML document easily with your editor. Simply open the file and use the edit features to modify the elements and structure of the file.  Lydia wants to add two new elements to her XML book file. She would like store the ISBN number and publication date of each book.

## Steps <sup>1234</sup>

1. Open the file **books.xml** in your text editor
2. Position the insertion point at the end of the **first Publisher element**, as shown in Figure N-15
3. Press **[Enter]** to insert a blank line, use tabs to align the insertion point directly beneath the beginning of the tag above, then type

**<ISBN>0517887835</ISBN>**

4. Press **[Enter]** again, align the insertion point, and type

**<Date>March 1998</Date>**

The text in the editor should now match Figure N-16. All that remains is to enter the new elements for the other two book records.

5. Insert the following elements in the two remaining book records:

**<ISBN>1558212485</ISBN>**      **<ISBN>0070653151</ISBN>**

**<Date>February 1994</Date>**      **<Date>October 1996</Date>**

The XML document should now contain the text shown in Figure N-17.

6. Save, then close the file

FIGURE N-15: Correct position for insertion point

```
<?XML VERSION='1.0'?>
<BookList>
  <Book>
    <Name>The Backpacker's Field Manual</Name>
    <Author>Rick Curtis</Author>
    <Publisher>Crown Publishers Inc.</Publisher>
  </Book>
  <Book>
    <Name>The Modern Backpacker's Handbook</Name>
    <Author>Glenn Randall</Author>
    <Publisher>Lyons and Burford Publishers</Publisher>
  </Book>
  <Book>
    <Name>The Backpacker's Handbook</Name>
    <Author>Chris Townsend</Author>
    <Publisher>Ragged Mountain Press</Publisher>
  </Book>
</BookList>
```

Insertion point at the end of the first Publisher element

FIGURE N-16: XML with two new elements

```
<?XML VERSION='1.0'?>
<BookList>
  <Book>
    <Name>The Backpacker's Field Manual</Name>
    <Author>Rick Curtis</Author>
    <Publisher>Crown Publishers Inc.</Publisher>
    <ISBN>0517887835</ISBN>
    <Date>March 1998</Date>
  </Book>
  <Book>
    <Name>The Modern Backpacker's Handbook</Name>
    <Author>Glenn Randall</Author>
    <Publisher>Lyons and Burford Publishers</Publisher>
  </Book>
  <Book>
    <Name>The Backpacker's Handbook</Name>
    <Author>Chris Townsend</Author>
    <Publisher>Ragged Mountain Press</Publisher>
  </Book>
</BookList>
```

ISBN element

Date element


FIGURE N-17: XML file modifications complete

```
<?XML VERSION='1.0'?>
<BookList>
  <Book>
    <Name>The Backpacker's Field Manual</Name>
    <Author>Rick Curtis</Author>
    <Publisher>Crown Publishers Inc.</Publisher>
    <ISBN>0517887835</ISBN>
    <Date>March 1998</Date>
  </Book>
  <Book>
    <Name>The Modern Backpacker's Handbook</Name>
    <Author>Glenn Randall</Author>
    <Publisher>Lyons and Burford Publishers</Publisher>
    <ISBN>1558212485</ISBN>
    <Date>February 1994</Date>
  </Book>
  <Book>
    <Name>The Backpacker's Handbook</Name>
    <Author>Chris Townsend</Author>
    <Publisher>Ragged Mountain Press</Publisher>
    <ISBN>0070653151</ISBN>
    <Date>October 1996</Date>
  </Book>
</BookList>
```

All records include new ISBN and Date elements



# Altering XML Data View with HTML

When you add elements to an XML document, you must make corresponding changes to the HTML document you are using to view the data; otherwise, the new data will not be displayed. Fortunately, it is easy to bring the HTML document into alignment with the new XML elements. Simply insert the code necessary to display the new elements in the format you desire.  Lydia decides to display the ISBN and publication date in the same table with the rest of her book-list data.

## Steps 1234

1. Open the file **books.htm** in your text editor
2. Change the font face for the Table header from **Arial** to **Arial Black** in the opening Table tag
3. Place the insertion point at the end of **<TH>PUBLISHER</TH>**
4. Press **[Enter]** to insert a blank line, tab over to align up directly beneath the tag above, then type **<TH>ISBN</TH>**
5. Press **[Enter]** to insert a blank line, align with the tag above, then type **<TH>DATE</TH>**

These tags will create two new column headings—ISBN and DATE—for the data table.

6. Place the insertion point at the end of the **last row in the table**, as shown in Figure N-18
7. Press **[Enter]**, align the insertion point directly under the beginning of the first tag above, then type **<TD VALIGN="top"><DIV DATAFLD="ISBN" DATAFORMATAS="HTML"></DIV></TD>**
8. Press **[Enter]**, align the insertion point, then type **<TD VALIGN="top"><DIV DATAFLD="DATE" DATAFORMATAS="HTML"></DIV></TD>**

The text in the editor should match Figure N-19. Check it carefully to make sure there are no typos.

9. Save the file, then close the document and text editor

### Trouble?

If your HTML document displays a parsing error, use your text editor to find the syntax mistake in the books.xml file, then refresh your browser screen.

10. Start your browser, then open the **books.htm** file to view the changes made to the table. The HTML document appears with the two new column headings and corresponding data, as shown in Figure N-20.

**FIGURE N-18:** Position insertion point to enter column data tags

```
<HTML>
<HEAD>
  <TITLE>Backpacking Book List</TITLE>
</HEAD>
<BODY>
  <H2>Recommended Backpacking Books</H2>

  <APPLET CODE="com.ms.xml.dso.XMLDSO.class" WIDTH="100%"
  HEIGHT="25" ID="xml_dso" MAYSCRIPT="true">
    <PARAM NAME="url" VALUE="books.xml">
  </APPLET>

  <TABLE ID="table" BORDER="2" WIDTH="100%"
  DATASRC="#xml_dso" CELLPADDING="5">
    <THEAD>
      <FONT FACE="Arial Black" SIZE="2">
        <TR>
          <TH>TITLE</TH>
          <TH>AUTHOR</TH>
          <TH>PUBLISHER</TH>
          <TH>ISBN</TH>
          <TH>DATE</TH>
        </TR>
      </FONT>
    </THEAD>
    <FONT FACE="Times New Roman" SIZE="2">
      <TR>
        <TD VALIGN="top"><DI> DATAFLD="NAME"
        DATAFORMATAS="HTML"></DI></TD>
        <TD VALIGN="top"><DI> DATAFLD="AUTHOR"
        DATAFORMATAS="HTML"></DI></TD>
        <TD VALIGN="top"><DI> DATAFLD="PUBLISHER"
        DATAFORMATAS="HTML"></DI></TD>
      </TR>
```

Insertion point at the end of the line

**FIGURE N-19:** HTML altered to display new XML data

```
<H2>Recommended Backpacking Books</H2>

<APPLET CODE="com.ms.xml.dso.XMLDSO.class" WIDTH="100%"
HEIGHT="25" ID="xml_dso" MAYSCRIPT="true">
  <PARAM NAME="url" VALUE="books.xml">
</APPLET>

<TABLE ID="table" BORDER="2" WIDTH="100%"
DATASRC="#xml_dso" CELLPADDING="5">
  <THEAD>
    <FONT FACE="Arial Black" SIZE="2">
      <TR>
        <TH>TITLE</TH>
        <TH>AUTHOR</TH>
        <TH>PUBLISHER</TH>
        <TH>ISBN</TH>
        <TH>DATE</TH>
      </TR>
    </FONT>
  </THEAD>
  <FONT FACE="Times New Roman" SIZE="2">
    <TR>
      <TD VALIGN="top"><DI> DATAFLD="NAME"
      DATAFORMATAS="HTML"></DI></TD>
      <TD VALIGN="top"><DI> DATAFLD="AUTHOR"
      DATAFORMATAS="HTML"></DI></TD>
      <TD VALIGN="top"><DI> DATAFLD="PUBLISHER"
      DATAFORMATAS="HTML"></DI></TD>
      <TD VALIGN="top"><DI> DATAFLD="ISBN"
      DATAFORMATAS="HTML"></DI></TD>
      <TD VALIGN="top"><DI> DATAFLD="DATE"
      DATAFORMATAS="HTML"></DI></TD>
    </TR>
  </FONT>
```

Font face for table headings changed to Arial Black

Inserted column headings

New attributes to bind data to columns

**FIGURE N-20:** Five-column table with new data

Recommended Backpacking Books

Successfully loaded XML from "file:///A:/books.xml"

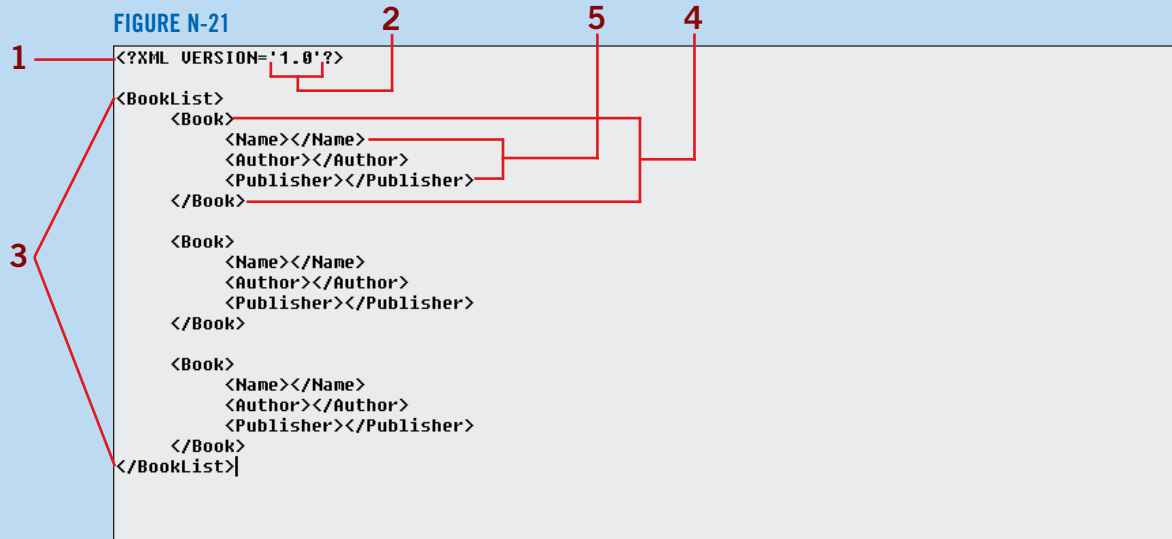
TITLE	AUTHOR	PUBLISHER	ISBN	DATE
The Backpacker's Field Manual	Rick Curtis	Crown Publishers Inc.	0517887835	March 1998
The Modern Backpacker's Handbook	Glenn Randall	Lyons and Burford Publishers	1558212485	February 1994
The Backpacker's Handbook	Chris Townsend	Ragged Mountain Press	0070653151	October 1996

New table headings

New row data

## ► Concepts Review

Label each item marked in Figure N-21.



Match each term with its description.

- |                     |  |
|---------------------|--|
| 6. Element          | a. An element that doesn't have a closing tag  |
| 7. Parser           | b. A text-based format that lets you describe, deliver, and exchange structured data |
| 8. XML              | c. No overlapping elements   |
| 9. Correctly nested | d. Dissects and interprets XML   |
| 10. Empty element   | e. The start and corresponding end tags, plus the contents in between the tagset     |

Select the best answer from the list of choices.

11. The XML declaration, in the prolog of an XML document, indicates that the document should be
- Encoded with XSL.
  - Processed as an XML file.
  - Rendered using HTML.
  - Displayed as HTML.
12. Which is *not* an XML syntax rule?
- All elements must have start and end tags.
  - All elements must be nested properly.
  - All attribute values must appear within quotation marks.
  - All empty elements must be terminated with `</>`.



**13. Which statement is False?**

- a. Blank space appearing in the content of an element is read as data.
- b. XML is organized in a tree-like structure.
- c. The elements `<Wine>red</Wine>` and `<wine>red</wine>` are equivalent in XML.
- d. XML brings structure and customization to the Web.

**14. Which is *not* a new table data binding attribute for HTML?**

- a. DATASRC
- b. DATAFLD
- c. DATAFORMAT
- d. DATAPAGESIZE

**15. Which one of the following is *not* required to display XML data in a browser?**

- a. parser
- b. Cascading Style Sheet (CSS)
- c. data-binding applet
- d. data-binding attributes

## ► Skills Review

**1. Define XML elements and structure.**

- a. Open your text editor, then type `<?XML VERSION='1.0'?>`
- b. Two lines down, enter the following elements, using the appropriate indentation to signify the structure of the document:

```

<Catalog>
  <Product>
    <Model></Model>
    <Price></Price>
    <Description></Description>
  </Product>

  <Product>
    <Model></Model>
    <Price></Price>
    <Description></Description>
  </Product>

  <Product>
    <Model></Model>
    <Price></Price>
    <Description></Description>
  </Product>
</Catalog>

```

- c. Verify your typing and correct any typos.
- d. Save the file as a text document with the filename `products.xml`.

**2. Enter XML data.**

- a. Make sure products.xml is open in your text editor.
- b. Position the insertion point between the first instance of the `<Model></Model>` element and type "Cleaner 100".
- c. Type "\$495" between the Price tags directly below the first Name element.
- d. Type "Domestic robot to vacuum and dust your home." between the Description tags.
- e. Enter the following content for the next two Product records:  
Whacker 300  
\$1,195  
Yard robot that mows and trims the edges of your lawn.  
  
Nursemaid 400  
\$4,995  
Personal robot to care for purchasers of Cleaner 100 and Whacker 200 robot models.
- f. Save and close the document.

**3. Bind XML data to HTML.**

- a. Start your text editor, then in a new document, type the following code with indicated alignments:  
`<HTML>`  
`<HEAD>`  
`<TITLE>Robot Product Catalog</TITLE>`  
`</HEAD>`  
`<BODY>`  
`<H2>Robot Product Catalog</H2>`
- b. Create two blank lines at the bottom the document, then bind the products.xml file to this HTML document by entering the following code:  
`<APPLET CODE="com.ms.xml.dso.XMLDSO.class" WIDTH="100%" HEIGHT="25"`  
`ID="xmlIdso" MAYSCRIPT="true">`  
`<PARAM NAME="url" VALUE="products.xml">`  
`</APPLET>`
- c. Type the following HTML closing tags:  
`</BODY>`  
`</HTML>`
- d. Check the document for errors, making changes as necessary.
- e. Save the file as a text document with the filename products.htm.

**4. Format XML data with HTML.**

- a. Make sure products.htm is open in your text editor.
- b. To create a table to bind, format, and display the data in the products.xml file, type the following HTML code just below the closing applet tag:  
`<TABLE ID="table" BORDER="2" WIDTH="100%" DATASRC="#xmlIdso" CELLPADDING="5">`  
`<THEAD>`  
`<FONT FACE="Arial" SIZE="2">`  
`<TR>`  
`<TH>MODEL</TH>`  
`<TH>PRICE</TH>`  
`<TH>DESCRIPTION</TH>`

```

</TR>
</FONT>
</THEAD>
<FONT FACE="Times New Roman" SIZE="2">
  <TR>
    <TD VALIGN="top"><DIV DATAFLD="MODEL" DATAFORMATAS="HTML"></DIV></TD>
    <TD VALIGN="top"><DIV DATAFLD="PRICE" DATAFORMATAS="HTML"></DIV></TD>
    <TD VALIGN="top"><DIV DATAFLD="DESCRIPTION" DATAFORMATAS="HTML"></DIV></TD>
  </TR>
</FONT>
</TABLE>

```

- c. Check the document for errors, making changes as necessary, then save and close the file.

### 5. Display XML data with HTML.

- Start your browser, then open the file products.htm.
- If you receive a parsing error, use your text editor to find and fix typos in either the products.htm or products.xml documents, then open the products.htm file again with your browser.
- When you are done examining the table of data, print the page, then close your browser.

### 6. Modifying an XML document.

- Open the file products.xml in your text editor.
- In the first Product element, insert the following new elements just below the Description element:
 

```

<Options>Turbo jet engine</Options>
<Delivery>2-4 weeks</Delivery>

```
- Enter the following elements for the last two records in your XML file:
 

```

<Options>Leaf collector</Options>
<Delivery>2-4 weeks</Delivery>

<Options>Medicine tray</Options>
<Delivery>4-6 months</Delivery>

```
- Check the document for errors, making changes as necessary, then save the file.

### 7. Alter XML data view with HTML.

- Open the file products.htm in your text editor.
- Insert the following HTML aligned below <TH>DESCRIPTION</TH>:
 

```

<TH>OPTIONS</TH>
<TH>DELIVERY</TH>

```
- Type
 

```

<TD VALIGN="top"><DIV DATAFLD="OPTIONS" DATAFORMATAS="HTML"></DIV></TD>
<TD VALIGN="top"><DIV DATAFLD="DELIVERY" DATAFORMATAS="HTML"></DIV></TD>

```
- Check the file for errors, making changes as necessary, then save the file and close your text editor.
- Open the products.htm file in your browser, and view the newly expanded table.
- If you receive a parsing error, use your text editor to find and fix the typing mistakes in either the products.htm or products.xml documents, then open the products.htm file once more with your browser.
- Print the document, then close your browser.

## ► Independent Challenges

**1.** You have just started buying music CDs and you would like to keep a list of them on your computer as your collection grows. You decide to use an XML file to store the name, song titles, and type of music for each music CD. At this point, you just want to create the custom elements and hierarchical structure of the XML document.

To complete this independent challenge:

- a. Open your text editor.
- b. Enter the prolog for an XML document, use the [Enter] key to create a couple of blank lines in the document, then type "<CDlist>".
- c. On the next line, press [Tab] once, then type "<CD>".
- d. On the next line down, press [Tab] twice, then type "<Name></Name>".
- e. On the next line down, press [Tab] three times, then type "<Song1></Song1>".
- f. Repeat Step 5 until you have entered tagsets for 10 songs (i.e., <Song2></Song2>...<Song10></Song10>).
- g. Below the last Song tagset, type "<Category></Category>".
- k. On the next line, press [Tab] once, then type "</CD>".
- i. Copy the <CD> element, and all its children elements, three times.
- j. At the bottom of the document, at the beginning of a new line, type "</CDlist>", then save the file as music list.xml.
- k. Print a copy of the document, then close your text editor.

**2.** Your best friend collects and sells comic books for a living. She asks you to help her create a computerized list of her collection that she can display on her personal Web site. You decide to use XML and HTML as the means of storing and displaying information about her comics. You use five records to test the design.

To complete this independent challenge:

- a. Open your text editor, then create an XML file with the root element <ComicList>.
- b. Add the parent element <ComicBook> and the children elements <Name></Name>, <Issue></Issue>, <Publisher></Publisher>, and <Value></Value>.
- c. Copy the ComicBook element and its children elements four times.
- d. At the bottom of the document, on a blank line, type "</ComicList>".
- e. Save the XML file as comics.xml.
- f. Enter the data from the table below into each ComicBook element in your XML document:
- g. Save the document, print it, then open a new document.
- h. Create an HTML document to display your XML data. Use the following code to bind the XML data to a table in your HTML file:

```
<APPLET CODE="com.ms.xml.dso.XMLDSO.class"
WIDTH="100%"
HEIGHT="25" ID="xmlDso" MAYSCRIPT="true">
<PARAM NAME="url" VALUE="comics.xml">
</APPLET>
```

```
<TABLE ID="table" BORDER="2" WIDTH="100%" DATASRC="#xmlDso" CELLPADDING="5">
<THEAD>
<TR>
<TH>TITLE</TH>
```

name	issue	publisher	value
Bombastic Five	4	Cool Comics	\$22,000
Radioactive Dog	12	Cool Comics	\$640
Sludge Man	34	Night Owl	\$26
Bombastic Five	7	Cool Comics	\$18,500
Sludge Man	19	Night Owl	\$35

```

<TH>ISSUE #</TH>
<TH>PUBLISHER</TH>
<TH>VALUE</TH>
</TR>
</THEAD>
<TR>
  <TD VALIGN="top"><DIV DATAFLD="NAME" DATAFORMATAS="HTML"></DIV></TD>
  <TD VALIGN="top"><DIV DATAFLD="ISSUE" DATAFORMATAS="HTML"></DIV></TD>
  <TD VALIGN="top"><DIV DATAFLD="PUBLISHER" DATAFORMATAS="HTML"></DIV></TD>
  <TD VALIGN="top"><DIV DATAFLD="VALUE" DATAFORMATAS="HTML"></DIV></TD>
</TR>
</FONT>
</TABLE>

```

- i. Format the table using the font style Arial with point size of 2.
- j. Save the file as a text document called comics.htm, then close your text editor.
- k. Open comics.htm in your browser to view your XML data. If the data fails to display, use your text editor to check your typing in comics.xml and comics.htm, and correct any typos.
- l. Print a copy of comics.htm from your browser.

**3.** You have been asked to inventory all the computers in your building at work and make the results available for viewing on your company's Intranet. Management would like to know the make, model, year, and location of each machine.

To complete this independent challenge:

- a. Create an XML file called computers.xml with the root element <Computers></Computers>.
- b. Add the parent element <Make></Make>, with the children elements <Model></Model>, <Year></Year>, and <Location></Location>. Copy the <Make></Make> element and its children elements five times.
- c. Populate the Model elements with data from the table below:
- d. Create an HTML file called computers.htm that will display the XML data in a table; use Arial as the font face.
- e. Open the HTML file in your browser.
- f. Print a copy of the document from your browser.

make	model	year	location
Mega Bite	T-Rex	1998	Office 101
Mega Bite	T-Rex	1998	Office 102
Mega Bite	Raptor	1998	Office 103
Tera Gig	Condor	1997	Office 104
Mega Bite	Raptor	1998	Reception Area



- 4.** You have decided to use XML and HTML to store and display a list of your favorite movies. Be sure to include custom elements (e.g., <MovieTitle></MovieTitle>) to store important information such as the name of the movie, your favorite actor, the plot, and other interesting data. Use the Internet to search for movie data if your information is not complete. In an XML file called movies.xml, structure the movie data so that each movie record is a child element of the parent element <MovieList></MovieList>. Display the data in a table with headings in an HTML file called movies.htm. Open the movies.htm file in your browser, then print the document.



## ► Visual Workshop

Create a Web application using XML and HTML to display the table of formatted data shown in Figure N-22. Print the document from your browser.

FIGURE N-22

### XML Products

Successfully loaded XML from "file:///A:/products.xml"

PRODUCT	COMPANY	DESCRIPTION
HoTMetaL Application Server	SoftQuad	Automates the database-to-XML and XML-to-HTML conversion process.
iNet Developer 4.0	Pictorius	Includes tools for parsing Document Type Definitions (DTD's) and converting HTML documents to XML.
XML Pro	Vervet Logic	A program that allows users to create and edit XML documents.